Network Controllability Perspectives on Graph Representation

Anwar Said, Obaid Ullah Ahmad, Waseem Abbas, *Member, IEEE*, Mudassir Shabbir, and Xenofon Koutsoukos

***Abstract*—Graph representations in fixed dimensional feature space are vital in applying learning tools and data mining algorithms to perform graph analytics. Such representations must encode the graph's topological and structural information at the local and global scales without posing significant computation overhead. This paper employs a unique approach grounded in networked control system theory to obtain expressive graph representations with desired properties. We consider graphs as networked dynamical systems and study their *controllability* properties to explore the underlying graph structure. The controllability of a networked dynamical system profoundly depends on the underlying network topology, and we exploit this relationship to design novel graph representations using controllability Gramian and related metrics. We discuss the merits of this new approach in terms of the desired properties (for instance, permutation and scale invariance) of the proposed representations. Our evaluation of various benchmark datasets in the graph classification framework demonstrates that the proposed representations either outperform (sometimes by more than 6%), or give similar results to the state-of-the-art embeddings.**

***Index Terms*—Graph classification, graph embedding, network controllability.**

I. Introduction

THE graph-theoretic framework provides means to analyze network characteristics and examine the influence of local interactions on global network behavior. In recent years, various data-driven approaches have been developed to solve real-world graph problems like graph classification, link prediction, community detection, and network evolution. Applying prevalent data mining techniques and learning algorithms to solve graph problems is not a straightforward task. The classical methods are designed for vector-valued data requiring graphs to be embedded in vector spaces. In other words, we need to define vector representations of graphs with some desired properties, such as permutation-invariance, expressiveness, and accuracy.

In this paper, we design a novel graph representation grounded in the *network controllability* paradigm [1]. We perceive graphs as *networked dynamical systems* in which each vertex is an *agent* (dynamical unit) that maintains a *state*. Every agent updates its state through some dynamical process and interacts with other agents in its neighborhood defined by the underlying network graph. The states of all agents define the overall network's state. The network controllability paradigm concerns steering a network from one state to another by injecting some *control signals* into the system through a subset of agents. The network's ability to be manipulated and controlled through such external inputs directly depends on the underlying network graph [2], [3], [4], [5], [6]. As a result, by studying the controllability properties of dynamical processes over networks, one can gather valuable insights into the underlying graph's structure that are distinct from other approaches. *We propose to understand and harness the relationship between graph topology and networked dynamical system behavior to design expressive graph representations in this work.*

The controllability of networked dynamical systems has been a fundamental topic in control theory. In recent years, many studies have established profound connections between network controllability and the underlying graph-theoretic constructs, such as matching [4], graph distances [7], dominating sets [8], equitable partitions [3], and zero forcing sets [9]. At the same time, graph-theoretic characterization of controllability for various families of network graphs, such as paths, cycles, trees, complete graphs, random graphs, symmetric graphs, circulant graphs, bipartite graphs, and product graphs, have been reported [1].

This paper demonstrates that by exploring controllability properties of networks, including how 'much' of the overall network can be controlled from a given set of input nodes, how 'easy' it is to steer the network towards desired states, how the 'location' of input nodes affect the controllability, and how the network topology influences these behaviors, we can construct effective graph representations (CTRL and CTRL+). We evaluate the proposed representations for the classification problem on several standard datasets and report improved or

Manuscript received 28 November 2022; revised 21 August 2023; accepted 28 October 2023. Date of publication 8 November 2023; date of current version 12 July 2024. This work was supported by National Science Foundation under Grants 2325416 and 2325417. Recommended for acceptance by B.C.M. Fung. *(Corresponding author: Anwar Said.)*

Anwar Said and Xenofon Koutsoukos are with Computer Science Department, Vanderbilt University, Nashville, TN 37235 USA (e-mail: anwar.said@vanderbilt.edu; xenofon.koutsoukos@vanderbilt.edu).

Obaid Ullah Ahmad is with the Electrical Engineering Department, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: obaidullah.ahmad@utdallas.edu).

Waseem Abbas is with the Systems Engineering Department, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: waseem.abbas@utdallas.edu).

Mudassir Shabbir is with Computer Science Department, Information Technology University, Lahore 54000, Pakistan, and also with Vanderbilt University, Nashville, TN 37235 USA (e-mail: mudassir.shabbir@vanderbilt.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2023.3331318, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2023.3331318

competitive classification accuracy compared to the existing approaches. We also discuss the expressiveness and invariance to node orderings of the proposed graph embeddings. This network control systems perspective to design graph representations is studied for the first time to the best of our knowledge.

II. Related Work

Graph representation methods can be broadly divided into three main approaches: *graph kernels*, spectral graph representations and *Graph Neural Networks (GNNs)*. In the following subsections, we provide a brief overview of the recent advancements in these domains.

Graph Kernels: Graph kernels are acknowledged as prevalent methods for analyzing and comparing graphs. There are a variety of widely recognized approaches, among them being the shortest-path kernel [10], the Weisfeiler Lehman kernel [11], the deep graph kernel [12], and the graphlet kernel [13], among others. These methods employ different graph-theoretic metrics like pairwise distances [10], subgraph mining [14], and neighborhood aggregation [11] to extract graph representations. To illustrate, the shortest-path kernel and its several variants, being some of the earliest and most impactful approaches, utilize the pairwise distance method to generate the kernel matrix [10]. The central concept revolves around comparing the lengths of the shortest paths between all possible pairs of vertices in two separate graphs. Similarly, random walk kernels offer an alternative viewpoint on graph kernels by assessing the count of walks within graphs that are mutual [15]. Another widely appreciated approaches involve interpreting the graph structure as a collection of vertices, mirroring the bag-of-words representation in textual contexts [13]. This approach, while overlooking the global perspective, zeroes in on the intricate details of the graph by examining the subgraph structures [16]. The Weisfeiler-Lehman (WL) kernel is another notable method in graph kernels. This technique distinguishes itself by going beyond the traditional graph structures, and instead, focusing on the neighborhood information of each node [11]. Despite the significant advances achieved by graph kernels, they do face certain limitations. For example, their performance may suffer when applied to large-scale graphs due to computational complexity. Additionally, their ability to capture more nuanced or complex patterns and structures within the graph data can be limited, as they mainly focus on local graph information [16].

Spectral Graph Representations: Unlike graph kernels, graph spectral methods leverage the spectrum (i.e., the eigenvalues) of the graph Laplacian, to describe graph structures [17]. This approach addresses the limitations of graph kernels by providing more computationally efficient solutions, especially for large-scale graphs, and capturing global patterns and structures within the data [18]. In recent years, various efforts have been undertaken that have garnered considerable interest. For instance, the authors in [17] introduce a family of graph spectral distances (FGSD), aiming to produce sparse and stable graph representations, with an emphasized focus on the uniqueness of the resulting graph representation. This proposed methodology leverages pairwise distances and considers the graph spectrum, comprising eigenvectors and eigenvalues, as a means to compute distances. In a similar vein, the authors of [19] propose the use of the Wasserstein distance to distinguish graphs based on their node feature distributions, while the distribution of smooth graph signals is employed in [20] for graph comparisons. An analogous approach, involving the optimal transport theory and discrete graph matching in a continuous domain, is proposed in [21], [22]. Similarly, [18] introduces a spectral graph representation approach, termed as NetLSD, which is based on the heat and wave kernel of the graphs. The crux of this work revolves around the notion of heat diffusion over the graph, assuming that the heat originates from a single vertex at time t and subsequently diffuses throughout the graph at different time scales. These time scales are computed in the form of a heat matrix, and the corresponding graph representation is obtained by taking the trace of the heat matrix calculated at each time instance. Furthermore, [14], [23] present graph descriptors that encapsulate the structural information of graphs using different graph-theoretic methods.

The proposed method of capturing the control-theoretic properties of a graph offers a rich perspective on obtaining graph representations. One way to understand such representation is by envisioning it as a heat diffusion process on graphs, similar to NetLSD. However, the diffusion in our proposal begins from multiple source nodes, offering a more comprehensive perspective for analyzing the graph structure. NetLSD is a special case of our proposed approach.

Graph Neural Networks (GNNs): GNNs are useful tools for learning graph representations. The last few years have seen a surge in GNNs approaches, introducing different techniques for improving models' capabilities. Among these, message-passing with attention, transformers, and more recently, unsupervised methods have significantly improved the models' performance on different tasks [24], [25]. Recently, various graph representation approaches have been introduced that focus on different aspects of the GNN methods i.e. scalability, robustness, generalizability, and explanability [26]. Such approaches include Graph Convolutional Networks [27], [28], Graph Reinforcement Learning [29], and Self-Supervised Learning (SSL) [30]. Graph convolutions involve two groups of methods: spectral and spatial convolution [31], [32]. The spectral convolutions methods use graph Fourier transform or its extensions to translate node representations to the spectral domain [31], [33]. On the other hand, spatial convolution methods use a message-passing mechanism to learn node embeddings [32], [34]. Reinforcement Learning (RL) based approaches implement RL mechanism to perform task-oriented learning on graphs such as graph generation, graph classification, and knowledge graph reasoning [35]. Self-Supervised Learning extracts informative knowledge through well-designed pretext tasks without relying on manual labels [30].

These graph neural network approaches report a state-of-the-art classification accuracy on several standard graph datasets, their sizable variance is of concern for certain applications [36]. Additionally, they utilize node features in the learning process, whereas the kernel methods usually work on unlabelled graphs.

Unlike the existing works, we pursue a unique approach to seek expressive graph representation. We consider graphs as networked dynamical systems and observe their controllability properties, revealing the extent to which a network can be manipulated. Using control theory, we employ tools to capture the relationship between networks' control behavior and their underlying topologies. We then propose a graph representation based on control properties that exhibit good classification accuracy for a broad range of datasets.

III. Networks as Dynamical Systems

A network of inter-connected entities is represented by a graph $G = (V, E)$, where the vertex set $V = V(G) = \{v_1, v_2, \ldots, v_N\}$ represents the entities, and the edge set $E = E(G) \subseteq V \times V$ represents the pairs of related entities. We use the terms vertex, node and agent alternatively. The neighborhood of a vertex v_i is the set $\mathcal{N}_i = \{v_j \in V : (v_j, v_i) \in E\}$. The degree of v_i, denoted by δ_i, is the size of the neighborhood $\mathcal{N}_i$. A graph with N nodes is represented by the adjacency matrix $J \in \{0,1\}^{N\times N}$, where $J_{i,j} = 1$, if and only if $(v_j, v_i) \in E$, and $J_{i,j} = 0$, otherwise. The degree matrix of G, denoted by D, is a diagonal matrix with $D_{i,i} = \delta_i$. The Laplacian matrix of a graph is defined as $L = D - J$. The transpose of a matrix X is denoted by X^T. An N-dimensional vector with all zero entries is denoted by $\mathbf{0}_N$, and a vector with all 1's is represented by $\mathbf{1}_N$. We consider undirected graphs here for the ease of exposition; however, all the methods and results are also applicable to directed graphs. We provide the details for directed graphs in the Supplementary material.

A. Problem Description

A graph embedding is defined as a function $\phi(G) : \mathcal{G} \to \mathbb{R}^d$, from the family of graphs, $\mathcal{G}$, to a d-dimensional Euclidean space. The objective of the graph embedding problem is to find *suitable* embeddings for the graphs, where the suitability of embeddings is driven by a few design goals discussed here. Most importantly, ϕ should be able to retain information about the *structural similarities* between pairs of graphs at both local and global scales, i.e., if two graphs are structurally similar, then their embeddings should generate vectors that are nearby with respect to the Euclidean distance in the target vector space. Note that the concept of similarities between two graphs is not a universal notion but rather depends on a particular application (e.g., graph classification, nearest neighbor search, clustering) and the family of graphs considered (e.g., chemical compounds, social networks). Furthermore, ϕ should be *permutation-invariant* in the sense that ϕ should return identical vectors for two graphs, G, H, with the same set of edges on a permuted vertex set, i.e., $\exists \pi : V(G) \to V(H), (u,v) \in E(G) \Leftrightarrow (\pi(u), \pi(v)) \in E(H)$, then we should have $\phi(G) = \phi(H)$. Another important design goal for graph embeddings is scale-adaptiveness. Not only should a graph embedding be able to map graphs of varying sizes to a fixed dimensional space, but mapping should also transcend the graph size to capture its structural properties. For example, an ideal graph embedding would map a cycle on ten nodes closer to the mapping of a cycle on twenty nodes as compared to the mapping of a wheel on fifteen nodes. In this paper, we address the problem of finding graph embedding while keeping the above-mentioned design goals in perspective.

B. Control Dynamics Properties Over Networks

We design distinctive graph representations by studying controlled dynamical processes over networks and mapping the control behavior to the network topology. Consider a network graph in which each agent v_i is a dynamical unit with a *state* $x_i(t) \in \mathbb{R}$ at time t that the agent also shares with its neighbors $\mathcal{N}_i$. Each agent updates its state by following some dynamics (e.g., consensus dynamics) while incorporating its neighbors' states during the state update process. The state of the overall system at time t is a vector of the states of all the agents, i.e., $x(t) = [x_1(t) \;\; x_2(t) \;\; \cdots \;\; x_N(t)]^T$. Each agent updates its state by the *consensus* dynamics given by

$$\dot{x}_i(t) = \sum_{v_j \in \mathcal{N}_i} (x_j(t) - x_i(t)). \tag{1}$$

The system level dynamics (evolution of the state $x(t)$) is then defined by the following linear system:

$$\dot{x}(t) = -Lx(t), \tag{2}$$

where L is the Laplacian matrix of the underlying network graph. It is well known that if G is connected, then state of each agent will eventually converge to the average of the initial states of all agents [1]. Thus, if $x_i(0)$ is the initial state (at $t = 0$) of agent v_i, then

$$x_i(t) \;\rightarrow\; \bar{x} \triangleq \frac{1}{N} \sum_{v_j \in V} x_j(0), \quad \text{as } t \to \infty, \tag{3}$$

$\forall v_i \in V$. It means the overall network state $x(t)$ will be $[\bar{x} \;\; \bar{x} \;\; \cdots \;\; \bar{x}]^T = \bar{x}\mathbf{1}_N \in \mathbb{R}^N$, as $t \to \infty$. Thus, under the consensus dynamics in (2), all agents converge to the same state. The linear system defined over G in (2) is autonomous as the system's state is updated without any external input. We have no control over the state's evolution in the sense that we cannot steer the system to some desired state, say $x^*(t_f) \in \mathbb{R}^N$ at time t_f. For this purpose, external control signals are injected into the system through a small subset of agents called *leaders*. Through these exogenous signals, leaders' states can be directly manipulated, i.e., $\dot{x}_l = u_l(t)$, where $u_l(t)$ is the input signal to the leader agent v_l. The non-leader agents, often called *followers*, continue to update their states using (1).

By feeding appropriate control signals to leaders, which are typically very few, the network's overall state $x(t) \in \mathbb{R}^N$ can be manipulated. As a result, we get certain control over the system's (state) evolution. The set of states that can be achieved, that is, to which the system can be driven, depends on the underlying network graph, the number of leaders, and their locations within the network. This ability of a network to be controlled through external inputs is called *network controllability*. By studying network controllability, we can gather valuable information about the network's structure as the two are deeply connected. By studying the control-related properties of the network, for instance, the number of leaders needed to completely control it, the dimension of the subspace consisting of controllable states, and the amount of control energy needed to steer the system from one state to another, one can thoroughly examine the graph structure and design effective graph representations.

IV. Network Controllability and Graph Representation

In this section, we will explore network controllability, a fundamental idea in network theory that helps us understand how we control systems. First, We will formally define a dynamic system for networks that can be controlled by probing the network using external signals. Second, we define network controllability and discuss various measures to quantify it. In the end, we will discuss the potential of these control-based measures in graph representation and illustrate their capability with a few examples. Based on these measures, we design novel embeddings for networks in Section V.

A. Network Dynamics

For a network graph $G = (V, E)$, we partition V into follower and leader nodes, denoted by V_f and V_ℓ, respectively, i.e., $V = V_f \cup V_\ell$. Here, $|V_f| = N_f$ and $|V_\ell| = N_\ell$. Without loss of generality, we assume that $V_f = \{v_1, v_2, \ldots, v_{N_f}\}$ and $V_\ell = \{v_{N_f+1}, \ldots, v_N\}$. The subgraph induced by V_f is called the *follower graph* and is denoted by G_f. The Laplacian matrix of G is partitioned as

$$L = \left[\begin{array}{c|c} A & B \\ \hline B^T & C \end{array}\right], \tag{4}$$

where $A \in \mathbb{R}^{N_f \times N_f}$, $B \in \mathbb{R}^{N_f \times N_\ell}$ and $C \in \mathbb{R}^{N_\ell \times N_\ell}$. An external input signal u_l is given to leader agent $v_l \in V_\ell$. The follower nodes update their states according to (1). The state vector corresponding to follower nodes is denoted by $\boldsymbol{x_f}(t) \in \mathbb{R}^{N_f}$, and is updated by the following system.

$$\dot{\boldsymbol{x}}_f(t) = -A\boldsymbol{x_f}(t) - Bu(t), \tag{5}$$

where A and B are in (4) and $u(t) = [u_{N_f+1}(t) \ \cdots \ u_N(t)]^T \in \mathbb{R}^{N_\ell}$ is a control signal at time t. We note that the system matrices $-A$ and $-B$ in (5) directly depend on the underlying network structure and the selection of leader agents. As a result, the evolution of $\boldsymbol{x_f}$ is a function of the network graph and the control mechanism, which includes external inputs and the selection of leader agents in the network. From the control perspective, we are interested in knowing if it is possible to steer the system (5) from an arbitrary initial state to an arbitrary final state in a finite amount of time t_1. If it is possible, then how much control energy $\mathcal{E}(u)$, defined below, would be required.

$$\mathcal{E}(u) = \int_{\tau=0}^{t_1} \|u(\tau)\|^2 d\tau. \tag{6}$$

Similarly, if all states are not reachable, then what is the dimension of the subspace consisting of reachable states? How these control properties vary as a result of a change in leader agents? Answers to these questions encode information that could be conducive to learn the graph structure. For this, we need metrics to quantify various aspects of network controllability. These measures can then be used to obtain graph embeddings. Controllability of linear systems is a fundamental topic in Control theory and we use results from there to quantify the controllability properties of networks.

B. Network Controllability Metrics

Controlling a network corresponds to driving a network from a given initial state to a desired final state by applying control inputs to leaders in the network. If $\boldsymbol{x_f}(t_i)$ is the initial state at time t_i, then under the dynamics (5), the state at time t_s is

$$\boldsymbol{x_f}(t_s) = e^{-A(t_s-t_i)}\boldsymbol{x_f}(t_i) + \int_{t_i}^{t_s} e^{-A(t_s-\tau)}(-B)u(\tau)d\tau. \tag{7}$$

A state $\boldsymbol{x}^*_{\boldsymbol{f}} \in \mathbb{R}^{N_f}$ is called *reachable* if there exists an input that can drive the network from origin $\mathbf{0}_{N_f}$ to $\boldsymbol{x}^*_{\boldsymbol{f}}$ in a finite amount of time. The set of all reachable states constitutes the *controllable subspace*.[1] The *dimension* of the controllable subspace is an important control-theoretic property and can be computed by the *rank* of the *Controllability matrix* below, [6].

$$\mathcal{C} = \begin{bmatrix} -B & (-A)(-B) & \cdots & (-A)^{N_f-1}(-B) \end{bmatrix}. \tag{8}$$

The rank of the above matrix depends on A and B, which in turn depend on the network graph and the selection of leaders. The network is *completely controllable* if and only if $rank(\mathcal{C}) = N_f$. *Controllability Gramian* is an important mathematical object that provides crucial information about the control behavior of the network [5], [37], [38]. Using controllability Gramian, we can quantify how 'easy' it is to go from one state to another in terms of the required control energy (6). For the system in (5), the *infinte horizon controllability Gramian* is defined as [5], [37],

$$\mathcal{W} = \int_0^\infty e^{-A\tau}(-B)(-B)^T e^{-A^T\tau}d\tau \in \mathbb{R}^{N_f \times N_f}. \tag{9}$$

If the system is stable, that is, all eigenvalues of $-A$ have negative real parts, then $\mathcal{W}$ converges asymptotically and can be computed by the Lyapunov equation,

$$(-A)\mathcal{W} + \mathcal{W}(-A)^T + (-B)(-B)^T = 0, \tag{10}$$

which is a system of linear equations and is therefore easily solvable. For the solution of (10) to exist, $-A$ must be a stable matrix, which is true for connected graphs.

Lemma 1: If we partition the Laplacian matrix L of an undirected connected graph as in (4), then the matrix A is positive definite [1].

As a result, $-A$ is negative definite in the case of connected graphs and the system in (5) is stable, and the corresponding $\mathcal{W}$ can be computed. Controllability Gramian provides an energy-related quantification of controllability, and we can obtain several controllability statistics from $\mathcal{W}$ [5], [37], [38]. We discuss some of them below.

[1] In continuous linear time-invariant systems, as in (5), if a state $\boldsymbol{x}^*_{\boldsymbol{f}}$ is reachable from the origin, then $\boldsymbol{x}^*_{\boldsymbol{f}}$ is also reachable from an arbitrary initial state in any duration of time.

- *Trace of $\mathcal{W}$:* The trace of the controllability Gramian is inversely related to the average control energy over random target states. It can also be considered as a measure of average controllability in all directions in the state space.
- *Minimum eigenvalue of $\mathcal{W}$:* It is the worst-case metric that is inversely proportional to the control energy required to steer the network in the least controllable direction in the controllable subspace.
- *Rank of $\mathcal{W}$:* The rank of $\mathcal{W}$ is the dimension of the controllable subspace.
- *Determinant of $\mathcal{W}$:* The quantity $\text{ld}(\mathcal{W}) = \log(\prod_j \mu_j(\mathcal{W}))$, where $\mu_j(\mathcal{W})$ is a non-zero eigenvalue of $\mathcal{W}$, is a volumetric measure of the controllable subspace reachable with one unit or less of control energy. If the system is completely controllable, then $\text{ld}(\mathcal{W})$ is the log determinant of $\mathcal{W}$.

C. Main Idea

In this section, we demonstrate the main idea regarding *how can we design successful graph embeddings by exploiting the behavior of controlled dynamical processes* over networks. The controllability metrics, as defined in the above section, have the capability to capture the underlying local and global network topology and merit to distinguish certain graph families. For instance, Wang in [39] proves that every graph G satisfying a constraint on the determinant of the *controllability matrix* $\mathcal{C}$ can be distinguished by the collective spectrum (eigenvalues) of the adjacency matrix $A \in \mathbb{R}^{N\times N}$ of graph G and its complement. Theorem 1 in [40] states:

Theorem 4.1: Every graph in family $\mathcal{F}_N$ can be determined completely by its generalized spectrum, where

$$\mathcal{F}_N = \left\{ G \mid \frac{det(\mathcal{C})}{2^{\lfloor \frac{N}{2} \rfloor}} \text{is an odd square-free integer} \right\} \tag{11}$$

Wang et al. extends this theorem and proves in [41] that the *tournament networks* belong to $\mathcal{F}_N$ and can be identified from the adjacency spectrum exhibiting the *usefulness of controllibility metrics in distinguishing graphs*. Similarly, there are numerous results in the literature providing insights into the relationship between the network topology and the controllability properties, thus, establishing the potential of controllability ideas to distinguish graphs in various families. Some simple examples to further illustrate the differentiating capabilities of the control metrics are presented below.

Examples: We illustrate through examples that network controllability depends on the topological organization of the network and the location of leaders in it. Fig. 1 shows various networks, each of which has eight agents, including a single leader agent. The controllability properties of the resulting follower networks, for instance, the rank and the trace of the controllability Gramian, denoted by **rank** and **tr**, respectively, vary in networks. The path network in Fig. 1(a) is completely controllable with a single leader agent, which is one of the end nodes. At the same time, the complete network in Fig. 1(c) is least controllable as the rank of the controllability Gramian is 1. Similarly, in other networks, the controllability attributes are

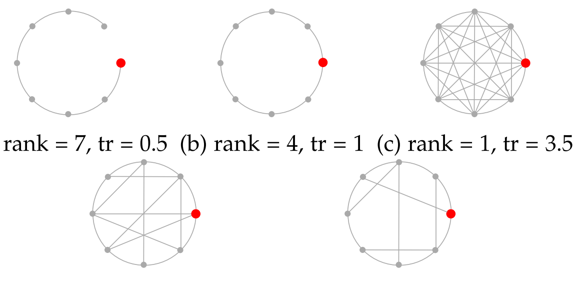

Fig. 1. Controllability metrics are functions of the underlying network graph.

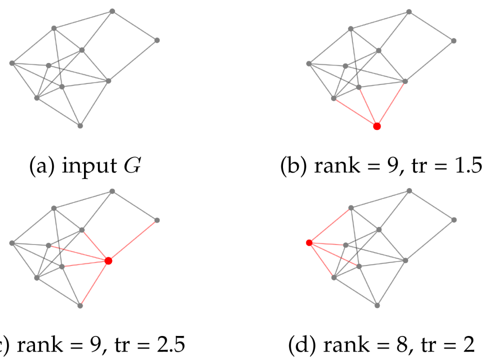

Fig. 2. Controllability metrics vary with leader selection

functions of the network graph. Along with the network topology, leader selection also affects the network controllability, as illustrated in Fig. 2. We consider a network of 10 agents, of which one is a leader agent, which means $N_f = 9$. In Fig. 2(b), the dimension of the controllable subspace is 9, which means that the follower network is completely controllable. The edges between a leader and follower nodes, which decide the structure of B matrix in (5) are shown in red. In Fig. 2(c), we choose a different leader and observe that the network remains completely controllable; however, the trace of $\mathcal{W}$ changes.

To collect valuable information about the graph structure, we need to probe the network effectively. This can be achieved by varying the number and locations of leader nodes and observing the resulting controllability behavior using measures $\mathbf{tr}(\mathcal{W})$, $\mu_j(\mathcal{W})$, $\mathbf{rank}(\mathcal{W})$ and $\mathbf{ld}(\mathcal{W})$. In the next section, we use these controllability metrics collected by various choices of leader selection to construct useful graph representations.

Remark 1: The consensus dynamics over the network can be defined in several other ways, for instance, by considering the overall state of the network (instead of only the followers' states $\boldsymbol{x_f}$) and therefore, selecting $-L$ as the system matrix.

We have discussed various controllability metrics, including the Gramian trace, eigenvalues, rank, and resolvability, which capture important characteristics of the graph in this section. As illustrated above, these measures provide valuable insights into the structure and dynamics of the corresponding graph. Now, armed with this knowledge, we will present how we can utilize these metrics to construct expressive graph representations in the next section.

V. Graph Representation Design

This section describes how we design our graph representation based on controllability characteristics defined in the previous section. If the graph is not connected, then the systems matrix $-A$ in (5) might not be stable for some choice of leader agents. Consequently, the solution of (10) might not exist. Therefore, we assume that the input network graph is connected. However, some real-world phenomena may generate networks that are disconnected. To handle such cases, we perform a preprocessing step in which we introduce a new vertex in such a graph and add an edge from this new vertex to all other vertices in the graph. This ensures that the graph becomes connected. Similarly, for extremely tiny graphs (with less than ten vertices), we perform a cloning step in which multiple copies of the original graph are generated to ensure that the input graph contains at least ten vertices. This is not an ideal solution as it tampers with the graph's structure, but we believe this preprocessing is rarely needed, and it retains enough structural information from the original graph, as we illustrate in the evaluation section.

We have two main probes to explore the controllability aspects of a network graph: the *number of leader agents* and their *locations* in the network. For molecular datasets, we utilize node information and the leader selection process is deterministic. For the rest of the datasets, the leader selection process is uniformly random and the features of the Gramian are calculated for several different number of leaders. The details of this leader selection process for each dataset are provided in Section VI.

For a given set of leader agents, we first compute the corresponding system matrices, i.e., $-A$ and $-B$ by partitioning the Laplacian matrix as in (4). Then, we compute the corresponding controllability Gramian $\mathcal{W}$ as in (10). We note that to solve the Lyapunov (10), efficient algorithms and solvers exist that scale well to large networks [42], [43]. We record the trace, rank, minimum (non-zero), and maximum eigenvalues of the Gramian. Along with these controllability measures, we also include a few easy-to-compute statistics about the input graph that are mentioned in Algorithm 1. A step-wise description of the embedding, which we call **CTRL**, is given in Algorithm 1. We provide further details of the measures we use in the embeddings in supplementary materials. We cater any structural changes to the input graph due to the preprocessing step by also designing an enhanced version, called **CTRL+**. In CTRL+, we simply concatenate the information in CTRL with another graph embedding that may have better expressiveness for disconnected and small graphs. For this work, we use a recent graph embedding called *Higher-Order Structure Descriptor (HOSD)*, which is easy to implement and contains the count of various small subgraphs present at multiple scales in the network [14]. Since CTRL is reasonably sized, concatenating with HOSD does not create any severe computational overheads.

Next, we analyze some of the properties of CTRL that are vital for the network classification task as discussed in Section III-A.

Permutation Invariance: The CTRL uses the spectrum (set of eigenvalues) of the controllability Gramian to calculate the feature descriptors of the given graph G. For a given graph G with N nodes and given leaders V_l, the Gramian is defined as:

Algorithm 1: CTRL: (Control Embedding).

```
Input: Graph $G = (V, E)$, $|V| = N$, num_iterations
Output: Graph embedding $\mathcal{R}$
1: $L \leftarrow$ Laplacian of $G$.
2: Initialize $\mathcal{R}$ as an empty list.
3: Create a list of leader sets using the leader selection
   strategy (see experimental setup).
4: for each $V_l$ in the list of leader sets do
5:    Compute the corresponding system matrices $-A, -B$
      (as in (5)).
6:    Compute the Gramian $\mathcal{W}$ (as in (9) and (10)).
7:    Compute the rank, trace, minimum non-zero
      eigenvalue, $\mu_{\min}$, and maximum eigenvalue, $\mu_{\max}$, of
      $\mathcal{W}$.
8:    Compute the Resolvability (number of unique
      distance-to-leader vectors from [7]) of $G$
      corresponding to selected leader nodes
9:    Compute rank, trace, $\mu_{\min}$, $\mu_{\max}$, Resolvability value
      and concatenate to the list $\mathcal{R}$.
10: end for
   (Adding some simple stats about the
   graph structure to $\mathcal{R}$.)
11: Concatenate $N$, $|E|$, number of bi-connected
   components, the Laplacian spectrum. (We add
   three smallest and largest
   eigenvalues), Laplacian energy, eccentricity
   spectrum, eccentricity energy, Wiener index, trace
   degree sequence of $G$, and cycles information.
12: Return $\mathcal{R}$
```

$$\mathcal{W}_G^{(V_l)} = \int_0^\infty e^{-A\tau}(-B_{V_l})(-B_{V_l})^T e^{-A^T\tau} d\tau \tag{12}$$

where $B_{V_l} \in \mathbb{R}^{N_f \times N_l}$ for V_l leaders. We show that when we select the leader nodes uniformly at random, the expectation of spectrum of the Gramian remains independent of the permutation of vertices in the adjacency and Laplacian matrices.

Proposition 5.1: The expected spectrum of the Gramian of follower dynamics (as in (12)) of a leader-follower network G, in which the leaders selection is uniform random, is permutation invariant.

Proof: For uniform random selection of leaders, the expectation of the Gramian matrix $\mathcal{W}_G^{(j)}$ is:

$$E[\mathcal{W}_G] = \frac{1}{\mathcal{K}_l}\sum_{j=1}^{\mathcal{K}_l}\int_0^\infty e^{-A\tau}(-B_{V_j})(-B_{V_j})^T e^{(-A\tau)^T} d\tau \tag{13}$$

where $\mathcal{K}_l = \binom{N}{N_l}$ is the number of choices for a leader set of size N_l, and B_{V_j} is the input matrix corresponding to V_j leader set. A permutation matrix Π is a square matrix, I, formed by a reordering of the rows of the identity matrix of the corresponding dimensions. If $G = (V, E)$ and $G' = (V', E')$ are isomorphic graphs, then there exists a permutation matrix Π

such that $\Pi\ L\ \Pi^{-1} = L'$, where L and L' are the corresponding Laplacian matrices of G and G' respectively. Recall that Π satisfies the property $\Pi\ \Pi^T = \Pi^T\Pi = I$. Let $\Pi_f \in \mathbb{R}^{N_f \times N_f}$ be an arbitrary permutation matrix corresponding to the follower agents. The Gramian $\mathcal{W}_{G'}$ of the follower dynamics obtained from a permuted copy of the Laplacian matrix $L' = \Pi L \Pi^{-1}$ is

$$\begin{aligned}\mathcal{W}_{G'}^{(j)} &= \int_0^\infty e^{-A'\tau}(-B'_{V_j})(-B'_{V_j})^T e^{(-A'\tau)^T} d\tau \\ &= \int_0^\infty e^{-\Pi_f A\tau\Pi_f^{-1}}(-B'_{V_j})(-B'_{V_j})^T e^{(-\Pi_f A\tau\Pi_f^{-1})^T} d\tau \\ &= \Pi_f \int_0^\infty e^{-A\tau}\Pi_f^T(-B'_{V_j})(-B_{V_j})^T \Pi_f e^{(A\tau)^T} d\tau \Pi_f^T\end{aligned}$$

where $\Pi_f^{-T} = (\Pi_f^{-1})^T = (\Pi_f^T)^T = \Pi_f$. Further, we can write,

$$= \Pi_f \left(\int_0^\infty e^{-A\tau}(\Pi_f^T B'_{V_j})(\Pi_f^T B'_{V_j})^T \Pi_f e^{(A\tau)^T} d\tau \right) \Pi_f^T$$

Note that the summation of expectation in (13) is over all choices of leader sets of size N_l. For each $(\Pi_f^T B'_{V_j})$, there is a unique $1 \leq j' \leq \mathcal{K}_l$ such that $(\Pi_f^T B'_{V_j}) = B_{V'_j}$. Therefore,

$$\begin{aligned} E[\mathcal{W}_{G'}] &= \frac{1}{\mathcal{K}_l} \sum_{j'=1}^{\mathcal{K}_l} \Pi_f \mathcal{W}_G^{(j')} \Pi_f^T \\ E[\mathcal{W}_{G'}] &= \Pi_f \left(E[\mathcal{W}_G]\right) \Pi_f^T. \end{aligned}$$

Thus, the expected Gramian of a permuted graph is same as the the permutation of the expected Gramian of the original graph. Since the spectrum of a matrix is invariant to linear transformations, we conclude that the expected spectrum of the Gramian is preserved under vertex permutations.

Moreover, a simple numerical analysis on Erdős-Rényi (ER) graphs shows that the CTRL descriptor converges towards the mean value with fairly low variance for uniformly random leader selection. We randomly generate 25 graphs with $N = 30$ nodes and the probability p of an edge between any two nodes to be 0.18. We plot the mean and variance of rank and trace of the Gramian for all the leaders combinations from 1 to 5 as shown in Figs. 3(a) and (b). We also generate 25 ER graphs with $N = 100$ and $p = 0.18$. For these larger graphs, we uniformly randomly select 50000 leaders sets for number of leaders from 1 to 30 and plot the mean and variance values of rank and trace of the Gramian shown in Figs. 3(c) and (d), respectively. Interestingly, the rank has very low variance for both graphs sizes. The variance of the trace value is relatively high for lower number of nodes (because of lower number of leaders combinations), but drops for higher number of leaders.

Scale Invariance: For certain families of graphs, some of the control descriptor features have the capability to be consistent with the variation of the size of the graphs. For instance, the Gramian is full rank when either terminal node of a path graph is selected as a leader [6]. Liu et al. in [44] provides a relation for $\mathbf{rank}(\mathcal{W})$ for path graphs when $N_l = 1$ and the leader is not a terminal node. Theorem 1 of [44] states that

Theorem 5.2: Suppose $P_N = (V, E)$ be the path graph where $V = \{v_1, \ldots, v_N\}$, and $(v_i, v_{i+1}) \in E\ \forall i = 1, \ldots, N$,

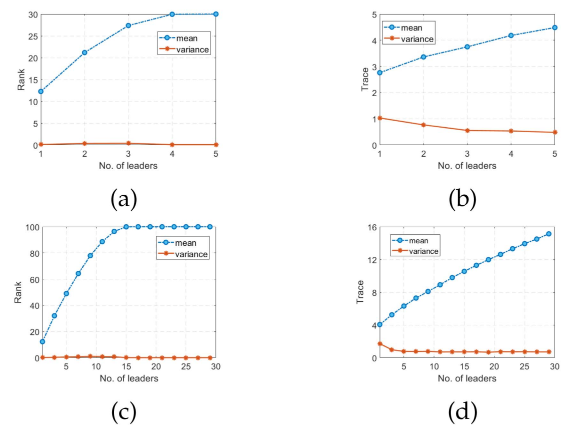

Fig. 3. (a) and (b) Show the mean and variance of the Rank and Trace obtained for the ER graphs with $N = 30$ for all combinations of leaders. (c) and (d) Show the same features for the ER graphs with $N = 100$ for uniformly randomly selected 50,000 leader sets.

and $\mathbb{F}_N = \{f_1, \ldots, f_m\}$, $f_1 > f_2 > \ldots > f_m > 1$, represent the set of the odd factors of N except for 1. If there exists a set M_i whose element m belongs to $\{2, 3, \ldots, N-1\}$ and f_i is the greatest common factor of $2m - 1$ and N, and if $j \in M_k$, where $k \in \{1, \ldots, m\}$, then the $\mathbf{rank}(\mathcal{W}) = N - (f_k - 1)/2$.

Hence, the $\mathbf{rank}(\mathcal{W})$ normalized by the size of the path graph N is fairly independent of the size of the path graphs. Likewise, for cycle graphs C_N with N nodes, the Gramian is strictly full rank if any two adjacent nodes are selected as leader nodes whereas for a single leader, the $\mathbf{rank}(\mathcal{W})$ is always $\lfloor N/2 \rfloor$ [4]. For the controllable subspace with two non-adjacent leaders in a cycle graph, the parity is important. Liu et al. in [44] discuss this controllability of a cycle with two leaders as a function of distances between the leaders. Theorem 4 of [44] states that

Theorem 5.3: Suppose $C_N = (V, E)$ be the cycle graph where $V = \{v_1, \ldots, v_N\}$, $(v_i, v_{i+1}) \in E\ \forall i = 1, \ldots, N$, $(v_1, v_N) \in E$. Let $I_k = \{mk | m \in \mathbb{N}_+\}$, and $I_k^n = \{x | x \in I_k\ \&\ x \leq \lceil N/2 \rceil\}$. For even N, let $\mathbb{F}_N^E = \{f_1, \ldots, f_m\}$ denote all even factors of N except for 2, where $N = f_1 > f_2 > \ldots > f_m > 3$ and let $F_1^E = I_{f_1/2}^N$, $F_k^E = I_{f_k/2}^N \backslash \cup_{i=1}^m F_i^E$ and $k = 2, \ldots, m$. Let $d(v_i, v_j)$ be the number of edges in the shortest path between v_i and v_j. If $d(v_1, v_2) \in F_k^E$, where $V_l = \{v_1, v_2\}$, then $\mathbf{rank}(\mathcal{W}) = N - f_k/2 + 1$.

The same procedure can be followed for odd N, thus, exhibiting the scale-invariance of the normalized $\mathbf{rank}(\mathcal{W})$ for cycle graphs to large degree.

VI. Experimental Evaluation

We evaluate the performance of the proposed embeddings on graph classification tasks and compare the results with the state-of-the-art graph representation methods. We consider classification accuracy as an evaluation metric and use 10-fold cross-validation in our experimental setting. We repeat the experiments ten times and report the mean of the best results of each iteration. CTRL and CTRL+embeddings are implemented in Python and all the experiments are performed on the Amazon Web Services instance $(c5.24xlarge)$ with 96-cores and 192 GB of RAM.

TABLE I
STATS OF THE DATASETS, INCLUDING THE NUMBER OF GRAPHS, AVERAGE NUMBER OF NODES AND EDGES, MINIMUM AND MAXIMUM NUMBER OF VERTICES, AND NUMBER OF CLASSES

Dataset	#Graphs	avg.$\lvert V\rvert$	avg.$\lvert E\rvert$	min.$\lvert V\rvert$	max.$\lvert V\rvert$	#Classes
MUTAG	188	17.93	19.79	10	28	2
PTC	344	14.29	14.69	2	109	2
PROTEINS	1113	39.06	72.82	4	620	2
ENZYMES	600	32.63	62.14	2	149	6
NCI1	4110	29.87	32.30	3	111	2
DD	1178	284.32	715.66	30	5748	2
IMDB-B	1000	19.77	96.53	12	136	2
IMDB-M	1500	13.00	65.94	7	89	3
REDDIT-B	2000	429.63	497.75	6	3782	2
REDDIT-M-5K	4999	508.52	594.87	22	3648	5

Datasets: We perform experiments on 10 standard graph classification benchmark datasets. MUTAG, PTC_MR, PROTEINS, ENZYMES, NCI1, and DD, are six bioinformatics datasets. IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, and REDDIT-MULTI-5 K are four social network datasets [45]. The bioinformatics datasets describe small molecules and chemical compounds that belong to two classes except for the ENZYMES dataset which consists of six classes. Among the social network datasets, IMDB-BINARY and IMDB-MULTI describe actors' ego-networks while REDDIT-BINARY and REDDIT-MULTI are chosen subreddits from an online social network. The graphs in IMDB-BINARY and REDDIT-BINARY are labeled with two classes and there are three and five classes in IMDB-MULTI and REDDIT-MULTI datasets, respectively. We provide basic stats of the datasets in Table I.

Source code and data including the precomputed CTRL embeddings on all the datasets are released to the research community for easier reproducibility of the results.[2]

Baselines: We consider six graph embeddings methods: Shortest Path (SP) [10], SVM theta [46], GK [47], NetSIMILE [48], NetLSD [18], and FGSD [17] for comparing the performance of the proposed method. Among them, the first *three are state-of-the-art* graph kernel methods while the later are *recently proposed graph descriptors*. NetLSD and FGSD use graphs' spectral features to extract graph information. NetSIMILE is a graph descriptor that is based on seven simple graph statistics including average vertex degree, average clustering coefficient, and standard deviation of the two-hops neighborhood. To evaluate the performance against state-of-the-art and recent GNN methods, we also consider nine Graph Neural Networks (GNNs) models: DGCNN [49], DiffPool [50], ECC [51], GIN [36], Nested Graph Neural Network (NGNN) with GIN [52], Cell Isomorphism Networks (CIN) [53], Message Passing Simplicial Networks (SIN) [54], Structural Semantic Readout (SSRead) [55] with SUM, and RepPool [56] for comparison. The GNNs models chosen for comparison include *well-known and recent models* showing promising results on the graph classification task.

Experimental setup:

To obtain the control features, we consider different leader selection techniques. The leader set selection process is as follows:

[2]https://github.com/Anwar-Said/Control-Graph-Embedding

- For molecular datasets i.e. MUTAG, PTC, and NCI1, we use node types for leaders selection process. We take all the nodes as leaders of the same type and repeat for all the distinct types of nodes in the dataset. For the ENZYMES dataset, the first node feature is considered as node type.
- For the rest of the datasets, the leader selection process is random. There are exponentially many choices to select N_l leaders among a set of N vertices. Therefore, we make this choice randomly. Formally, we uniformly select N_l leader nodes from the vertex set and repeat this random selection process c times for a given number of leaders. We first consider a fixed number of leaders, i.e., 1, 2, 5, 9, and then consider a fraction of overall nodes to be leaders, that is, 2%, 5%, 10%, 20%, 30% of the total nodes. For each random leader sets choice, we repeat the experiment 30 times. Every time a set of leader nodes is selected, we record graph measurements presented in the Algorithm 1. We use minimum, maximum, and average values of these measures over c iterations in our graph embedding.

We ensure through a preprocessing step that each graph has at least 10 nodes and is connected. We use the Random Forest (RF) algorithm with grid search for the classification and reported $10-$fold cross validation accuracy and their standard deviation. In the hyper-parameter setting of RF, we choose $\sqrt{d}$ features for building a tree, where d is the feature vector's size, and the classical Gini impurity is used as a metric to build the tree. The number of estimators is chosen from $\{50, 100, 500\}$ and total number of samples for a split in a tree is chosen from the set $\{2, 3, 4, 5\}$. In FGSD experiments, we set 0.0001 bin-width as recommended in [18]. For NetLSD, we use all variants mentioned in their paper and report the best results by utilizing the entire eigenspectrum. For NetSIMILE, we use their publicly available source codes and reproduce the results using our experimental setup. For graph kernel results, we use GraKel [57] library for computing kernel matrices and then use RF with the same setting for the classification. For GNNs, we reproduced the result of NGNN, CIN [53], SIN [54], GCN-SSRead [55] and RepPool [56] with 10-fold cross validation using the source codes made publicly available by the authors. The hyper-parameters are same as in the original papers. Here, we would like to note that due to the unavailability of a few datasets in the desired formats, we were not able to reproduce some of the results, hence N/A is reported. For DGCNN [49], DiffPool [50], ECC [51] and GIN [36], we consider the results in [58] that are obtained through identical frameworks.

Classification results:

We present the classification results of our evaluation in Tables II and III. We conclude from the results that the proposed embeddings either outperform or achieve comparable performance in terms of prediction accuracy on all benchmarks. Specifically, in comparison to the embedding methods, CTRL and CTRL+rank top on eight out of ten datasets. On the remaining datasets, the results are within 2% of the top results. In comparison to GNN models in Table III, we observe the superior performance of both CTRL and CTRL+on ENZYMES and REDDIT-B datasets. On the remaining datasets except

TABLE II
GRAPH CLASSIFICATION ACCURACY COMPARISON OF CTRL AND CTRL+ AGAINST SPECTRAL AND STATISTICAL GRAPH REPRESENTATION METHODS

Dataset	**SP** [10]	**SVM theta** [46]	**GK** [47]	**NetSIMILE** [48]	**NetLSD** [18]	**FGSD** [17]	**CTRL**	**CTRL+**
MUTAG	87.28	75.06	79.8	84.63	85.28	**88.07**	90.99	89.94
PTC	62.50	58.40	64.87	59.09	61.19	58.91	65.70	**62.78**
PROTEINS	72.68	68.64	72.51	71.15	**74.63**	70.18	75.2	74.76
ENZYMES	36.33	17.50	34.33	42.10	**44.25**	33.95	69.67	65.17
NCI1	68.69	50.68	61.22	73.96	76.31	**79.60**	81.31	81.8
DD	77.51	58.66	71.54	74.72	**77.27**	76.60	76.23	78.19
IMDB-B	72.40	50.0	76.60	74.41	73.79	73.88	**74.8**	75.0
IMDB-M	48.73	41.13	**49.66**	49.32	50.57	50.55	48.0	49.0
REDDIT-B	85.30	50.4	$> D$	**89.53**	89.12	81.60	95.9	96.05
REDDIT-M-5K	44.95	20.0	$> D$	52.78	53.58	OME	**52.91**	54.69

Top three results are highlighted by First, Second and Third. OME is out of memory and $> D$ indicates computations exceeds 24 hours.

TABLE III
GRAPH CLASSIFICATION ACCURACY WITH STANDARD DEVIATION COMPARISON AGAINST GNNS

datasets	**DGCNN**	**DiffPool**	**ECC**	**GIN**	**NGNN**	**CIN**	**SIN**	**SSRead**	**RepPool**	**CTRL**	**CTRL+**
MUTAG	85.83 ± 1.66	86.14 ± 10.99	N/A	90.00 ± 8.8	87.30 ± 5.6	90.00 ± 6.4	**90.55** ± 8.6	85.14 ± 4.5	92.12 ± 7.9	90.99 ± 6.26	89.94 ± 6.19
PTC	58.59 ± 2.47	67.26 ± 5.12	N/A	63.1 ± 5.7	55.20 ± 8.9	61.18 ± 9.8	66.21 ± 6.32	55.79 ± 5.6	71.34 ± 6.2	**65.70** ± 4.90	62.78 ± 4.82
ENZYMES	38.9 ± 5.7	59.5 ± 5.6	29.5 ± 8.2	**59.6** ± 4.5	33.0 ± 6.1	N/A	N/A	43.83 ± 8.8	50.76 ± 7.9	69.67 ± 4.8	65.17 ± 5.9
NCI1	76.4 ± 1.7	76.9 ± 1.9	76.2 ± 1.4	80.0 ± 1.4	78.6 ± 1.2	84.13 ± 1.6	80.26 ± 1.8	82.70 ± 1.84	79.53 ± 3.4	81.31 ± 2.1	**81.8** ± 1.6
PROTEINS	72.9 ± 3.5	73.7 ± 3.5	72.3 ± 3.4	73.3 ± 4.0	74.1 ± 3.7	73.24 ± 5.5	76.30 ± 4.0	73.76 ± 4.8	77.78 ± 3.7	**75.2** ± 3.1	74.76 ± 2.9
DD	**76.6** ± 4.3	75.0 ± 3.5	72.6 ± 4.1	75.3 ± 2.9	76.0 ± 4.3	N/A	N/A	70.23 ± 1.0	79.77 ± 4.8	76.23 ± 3.0	78.19 ± 2.2
IMDB-B	69.2 ± 3.0	68.4 ± 3.3	67.7 ± 2.8	71.2 ± 3.9	72.0 ± 4.2	73.90 ± 6.0	75.5 ± 3.8	71.4 ± 2.0	73.93 ± 4.7	**74.8** ± 2.3	75.0 ± 2.6
IMDB-M	45.6 ± 3.4	45.6 ± 3.4	43.5 ± 3.1	48.5 ± 3.3	**50.9** ± 3.4	50.93 ± 2.6	52.06 ± 3.4	48.66 ± 2.8	47.03 ± 3.8	48.0 ± 3.9	49.0 ± 3.7
REDDIT-B	87.8 ± 2.5	89.1 ± 1.6	OME	89.9 ± 1.9	$> 3D$	92.4 ± 2.1	93.0 ± 5.9	**93.8** ± 3.5	90.58 ± 4.5	95.9 ± 1.3	96.05 ± 1.5
REDDIT-5K	49.2 ± 1.2	53.8 ± 1.4	OME	56.1 ± 1.7	$> 3D$	31.8 ± 3.1	57.09 ± 1.7	53.25 ± 2.0	N/A	52.91 ± 1.7	**54.69** ± 2.2

The top three results are highlighted by First, Second and Third. OME is out of memory and N/A is the unavailability of the reproduced results (see Section 6). > 3D indicates training time exceeds 3 days.

IMDB-M, CTRL and CTRL+achieve second or third place. Although the proposed descriptor did not place in the top three on the IMDB-M dataset, the results are within 3% of the top results. These results clearly demonstrate the effectiveness of the proposed descriptor for graph classification.

We expect the results to further improve when combined with existing graph embeddings (spectral, statistical or GNNs). Our results empirically confirm that the spectral information of the network Gramian is effective for constructing graph representations.

VII. DISCUSSION

Our results reveals that the proposed control theoretic approach demonstrates promising results within the scope of both bioinformatics and social networks. This work represents a novel advancement, as it effectively combines control theory with graph machine learning, thereby offering a potent avenue for the enrichment of the graph representation domain. In the forthcoming sections, we delve into the computational complexity of the proposed method, as well as address the issues of scalability and randomness in the selection of leader nodes. Moreover, we also discuss a potential use case of CTRL embeddings for Self-Supervised Learning (SSL), which as an interesting avenue for future research.

Computational Complexity: We provide the analysis of the time complexity for each feature in Table I of the Supplementary material. Initially, we compute the system matrices, which involve array slicing with a complexity of $\mathcal{O}(N^2)$ where N is the number of nodes in the graph. Subsequently, the CTRL features, including the Gramian matrix and its properties, are computed for each set of leaders (as depicted in the for loop in line 4 of the algorithm). The computation of the Gramian matrix entails solving the generalized Sylvester equation and has a complexity of $\mathcal{O}(N^3)$ for sparse matrices [59]. For dense matrices, there are several iterative methods to compute the Gramian matrix [60]. One widely used method is the Bartels-Stewart method [61]. Its complexity is $\mathcal{O}(f(\sigma) \times N^3)$ where $f(\sigma)$ is a linear function of σ, the average number of iterations to ensure solution convergence. The computation of eigenvalues and other matrix properties also has a complexity of $\mathcal{O}(N^3)$. The calculation of resolvability also exhibits a time complexity of $\mathcal{O}(N^3)$. Therefore, the total computational cost of features for each set of leaders/followers is generally $\mathcal{O}(K \times f(\sigma) \times N^3)$, where K represents the number of distinct leader sets for each graph. For bioinformatics datasets, K is the number of vertex types in the whole dataset. Whereas, for the social network datasets, $K = c \times 9$ where $c = 30$ as explained in the experimental setup. The simple statistics about the graph structure have a time complexity of $\mathcal{O}(N^3)$. These statistics are independent of the

TABLE IV
TOTAL TIME TAKE TO COMPUTE CTRL EMBEDDINGS

Dataset	#Graphs	avg.$\|V\|$	Total Time
MUTAG	188	17.93	1.61 sec
PTC	344	25.56	1.81 sec
PROTEINS	1113	39.06	308 sec
ENZYMES	600	32.63	4.25 sec
NCI1	4110	29.87	92.79 sec
DD	1178	284.30	20.6 hrs
IMDB-B	1000	19.77	7.45 sec
IMDB-M	1500	13.00	6.21 sec
REDDIT-B	2000	429.61	5.623 hrs
REDDIT-M-5K	4999	508.50	8.231 hrs

CTRL features and are computed only once for the entire graph. The overall time complexity of Algorithm 1 can be expressed as:

$$\begin{aligned}&\mathcal{O}(K \times f(\sigma) \times N^3) + \mathcal{O}(N^3) + \mathcal{O}(N^3)\\&= \mathcal{O}(K \times f(\sigma) \times N^3)\end{aligned}$$

We present the timing information for computing the CTRL features on various datasets in Table IV. We note that the DD dataset has three graphs of sizes 2495, 4152, and 5749 that take a total time of 19.32 hrs to compute the CTRL feature embeddings. The CTRL features for the remaining 1175 graphs are computed in about an hour.

Computing CTRL features requires computing the controllability Gramian of the network control system defined on the graph. This step is computationally the most expensive and requires solving a matrix equation called the Sylvester equation. There are several approaches that can efficiently approximate the Sylvester equation. These approximations enable the proposed method to scale effectively on large graphs. We discuss some of them in more detail in the next subsection.

Scalability – The major step involved in the computation of the proposed representations is the computation of the infinite horizon controllability Gramian (9). Since it requires matrix multiplication (10), the overall time complexity is super-quadratic in N_f. Though it was not an issue for the graph classification tasks considered in the paper, the method could pose computational challenges for very large graphs.

We note that the scalability issue was not the main focus of this work, instead, a new approach relying on the control behavior of networks through external perturbations to encode graph structure was the main consideration. Nonetheless, there are several ways to deal with it and would be included in future extensions. Lyapunov equation, whose solution is the controllability Gramian, is a particular case of a more general Sylvester equation. While solving the Sylvester equation through standard methods (e.g., Bartels-Stewart, Hammarling) could be computationally expensive in large networks, several techniques have been developed over the years to significantly improve the computation time for approximately solving Lyapunov equations (LE) with reasonable accuracy. These techniques utilize the additional structure of LE, which includes the low-rank condition of the matrix (as the number of leaders is typically quite small), stability and sparsity of the system matrix, and so on. Some of these methods include iterative methods (e.g., cyclic low-rank Smith method), alternating directions implicit (ADI) methods, Krylov subspace methods, projection methods (e.g., extended Arnoldi or Glarekin method), see [62] and the references therein.

Leader Selection Mechanism –In the proposed work, we mainly considered random leader selection, albeit there can be other systematic ways; one being leader selection based on node types used for molecular datasets. Optimizing leader selection to maximize the performance for the task at hand could pose a significant computation overhead as leader selection problems are typically computationally challenging. For instance, it is NP-hard to determine if a graph with a fixed number of leader nodes is completely controllable or not. Thus, we desire a simple and computationally efficient scheme that gives good performance for a wide range of applications.

We note that in the control framework, leader nodes provide a mechanism to probe the network externally so that we can record the network control behavior and use it for graph embedding. In the absence of optimal leader selection, it is a reasonable proposition to probe the network fairly from all directions to achieve this objective. Random leader selection achieves these objectives, that is, efficient computation and fair probing of the network. However, improved results can be expected with a more standardized leader selection. We note an increase of about 10% accuracy in *NCI1* dataset with a systematic leader selection process defined in Section VI. It would be an interesting question to devise an optimal leader selection problem for specific tasks and study rigorously the *trade-off between the accuracy and computational costs.*

Self-Supervised Learning – Self-supervised learning (SSL) has emerged as a promising solution for challenging data labeling scenarios, offering a novel approach to address the problem of limited labeled data. SSL harnesses the inherent structure and unique characteristics of data to craft valuable representations without the need for explicit labels. Contrastive learning has been a popular approach in SSL, effectively extracting meaningful representations from unlabeled data by comparing positive and negative samples. For graph representation learning, Graph Contrastive Learning (GCL) has recently gained attention, maximizing agreement between similar nodes and informative embeddings capturing the graph structure [63]. Although existing graph contrastive learning approaches mainly focus on node-level embeddings, our proposed CTRL representation can have the potential for graph-level embeddings to be used for SSL.

Following the successful trend of contrastive learning in several other fields [63], [64], [65], [66], we believe that the CTRL embeddings has a potential to be used for learning meaningful graph embeddings with contrastive loss. Using the CTRL embeddings, we can train an encoder by maximizing agreement between learned embeddings of the original graph G and their augmented version G' in a new embedding space. The augmented version of a graph can be obtained using various transformations including complementing, node dropping, edge perturbation, subgraph sampling, etc [63]. As the control properties of a graph heavily depend on the topology of the graphs, one possibility would be to use the complement of the graph as the appropriate transformation since it mitigates any randomness during the transformation. By computing CTRL embeddings for both the original graph G and its transformed counterpart G', we can generate positive pairs, and the transformed counterpart of every other graph in a dataset will serve as a negative sample. The aim is to learn representations in a new embedding space that minimizes the InfoNCE loss [67], capturing meaningful graph representations effectively using CTRL embeddings of these positive and negative pairs. Overall, *the incorporation of CTRL embeddings into SSL provides a novel approach for addressing graph classification tasks* and we aim to pursue it further in future works.

VIII. Conclusion and Future Work

This work proposes that the networked dynamical system perspective, in particular, the network controllability paradigm offers a unique approach to encode the network structure for representing graphs. There are several directions to advance the controllability framework for graph representations. For instance, instead of Laplacian dynamics, one can consider different dynamical processes over networks. Similarly, we can use other controllability notions, such as, structural controllability, output, or target controllability, which concerns controlling a focused set of (target) nodes instead of the entire network [68], [69]. Also, there are alternative metrics that can be used to express the network's dynamical behavior, for instance, control centrality [70], Gramian-based edge centrality, control range index [71], and others (e.g., [38], [72]). Network control and graph learning communities share several scientific grounds, and viewing graph learning problems from the lens of control theory offers fresh perspectives and approaches to advance the field.

References

[1] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[2] S. Ahmadizadeh, I. Shames, S. Martin, and D. Nešić, "On eigenvalues of laplacian matrix for a class of directed signed graphs," *Linear Algebra its Appl.*, vol. 523, pp. 281–306, 2017.

[3] M. Egerstedt, S. Martini, M. Cao, K. Camlibel, and A. Bicchi, "Interacting with networks: How does structure relate to controllability in single-leader, consensus networks?," *IEEE Control Syst. Mag.*, vol. 32, no. 4, pp. 66–73, Aug. 2012.

[4] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, no. 7346, pp. 167–173, 2011.

[5] F. Pasqualetti, S. Zampieri, and F. Bullo, "Controllability metrics, limitations and algorithms for complex networks," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 40–52, Mar. 2014.

[6] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt, "Controllability of multi-agent systems from a graph-theoretic perspective," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 162–186, 2009.

[7] A. Yazıcıoğlu, W. Abbas, and M. Egerstedt, "Graph distances and controllability of networks," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 4125–4130, Dec. 2016.

[8] J. C. Nacher and T. Akutsu, "Analysis of critical and redundant nodes in controlling directed and undirected complex networks using dominating sets," *J. Complex Netw.*, vol. 2, no. 4, pp. 394–412, 2014.

[9] N. Monshizadeh, S. Zhang, and M. K. Camlibel, "Zero forcing sets and controllability of dynamical systems defined on graphs," *IEEE Trans. Autom. Control*, vol. 59, no. 9, pp. 2562–2567, Sep. 2014.

[10] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Proc. IEEE Int. Conf. Data Mining*, 2005, pp. 74–81.

[11] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, 2011.

[12] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1365–1374.

[13] R. Kondor, N. Shervashidze, and K. M. Borgwardt, "The graphlet spectrum," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 529–536.

[14] A. Ahmed, Z. R. Hassan, and M. Shabbir, "Interpretable multi-scale graph descriptors via structural compression," *Inf. Sci.*, vol. 533, pp. 169–180, 2020.

[15] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Proc. 16th Annu. Conf. Learn. Theory*, Washington, DC, USA, 2003, pp. 129–143.

[16] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Appl. Netw. Sci.*, vol. 5, no. 1, pp. 1–42, 2020.

[17] S. Verma and Z.-L. Zhang, "Hunt for the unique, stable, sparse and fast feature learning on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 88–98.

[18] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller, "NetLSD: Hearing the shape of a graph," in *Proc. 24th SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2347–2356.

[19] M. Togninalli, M. E. Ghisu, F. Llinares-López, B. Rieck, and K. M. Borgwardt, "Wasserstein weisfeiler-lehman graph kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[20] H. P. Maretic, M. E. Gheche, G. Chierchia, and P. Frossard, "GOT: An optimal transport framework for graph comparison," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[21] R. Flamary, N. Courty, A. Rakotomamonjy, and D. Tuia, "Optimal transport with laplacian regularization," in *Proc. Workshop Optimal Transport Mach. Learn.*, 2014.

[22] T. Yu, J. Yan, Y. Wang, W. Liu, and B. Li, "Generalizing graph matching beyond quadratic assignment model," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 853–863.

[23] A. Said, S.-U. Hassan, S. Tuarob, R. Nawaz, and M. Shabbir, "DGSD: Distributed graph representation via graph statistical properties," *Future Gener. Comput. Syst.*, vol. 119, pp. 166–175, 2021.

[24] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[25] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[26] A. Negro, *Graph-Powered Machine Learning*. New York, NY, USA: Simon and Schuster, 2021.

[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[28] S. Zhang and L. Xie, "Improving attention mechanism in graph neural networks via cardinality preservation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2020, Art. no. 1395.

[29] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6410–6421.

[30] Y. Liu et al., "Graph self-supervised learning: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 5879–5900, Jun. 2023.

[31] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*.

[32] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*., 2017, pp. 1024–1034.

[33] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv: 1710.10903*.

[35] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.

[36] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2018, *arXiv: 1810.00826*.

[37] T. H. Summers, F. L. Cortesi, and J. Lygeros, "On submodularity and controllability in complex dynamical networks," *IEEE Trans. Control Netw. Syst.*, vol. 3, no. 1, pp. 91–101, Mar. 2016.

[38] E. Wu-Yan, R. F. Betzel, E. Tang, S. Gu, F. Pasqualetti, and D. S. Bassett, "Benchmarking measures of network controllability on canonical graph models," *J. Nonlinear Sci.*, vol. 30, pp. 2195–2233, 2018.

[39] W. Wang, "Generalized spectral characterization of graphs: Revisited," *Electron. J. Combinatorics*, vol. 20, no. 4, 2013, Art. no. P4.

[40] W. Wei, "A simple arithmetic criterion for graphs being determined by their generalized spectra," *J. Combinatorial Theory, Ser. B*, vol. 122, pp. 438–451, 2017.

[41] W. Wang and L. Qiu, "Spectral characterizations of tournaments," *Discrete Math.*, vol. 345, no. 8, 2022, Art. no. 112918.

[42] J.-R. Li and J. White, "Low rank solution of lyapunov equations," *SIAM J. Matrix Anal. Appl.*, vol. 24, pp. 260–280, 2002.

[43] B. Vandereycken and S. Vandewalle, "A riemannian optimization approach for computing low-rank solutions of lyapunov equations," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 5, pp. 2553–2579, 2010.

[44] X. Liu and Z. Ji, "Controllability of multiagent systems based on path and cycle graphs," *Int. J. Robust Nonlinear Control*, vol. 28, no. 1, pp. 296–309, 2018.

[45] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *Proc. Workshop Graph Representation Learn. Beyond*, 2020.

[46] F. Johansson, V. Jethava, D. Dubhashi, and C. Bhattacharyya, "Global graph kernels using geometric embeddings," in *Int. Conf. Mach. Learn.*, 2014, pp. 694–702.

[47] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2009, pp. 488–495.

[48] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos, "Network similarity via multiple social theories," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2013, pp. 1439–1440.

[49] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI Conf. Artif. Intell.*, 2018.

[50] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," arXiv preprint 2018, *arXiv: 1806.08804*.

[51] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 29–38.

[52] M. Zhang and P. Li, "Nested graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021.

[53] C. Bodnar et al., "Weisfeiler and lehman go cellular: CW networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021.

[54] C. Bodnar et al., "Weisfeiler and lehman go topological: Message passing simplicial networks," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1026–1037.

[55] D. Lee, S. Kim, S. Lee, C. Park, and H. Yu, "Learnable structural semantic readout for graph classification," in *Proc. IEEE Int. Conf. Data Mining*, 2021, pp. 1180–1185.

[56] J. Li, Y. Ma, Y. Wang, C. Aggarwal, C.-D. Wang, and J. Tang, "Graph pooling with representativeness," in *Proc. IEEE Int. Conf. Data Mining*, 2020, pp. 302–311.

[57] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and M. Vazirgiannis, "GraKel: A graph kernel library in python," *J. Mach. Learn. Res.*, vol. 21, no. 54, pp. 1–5, 2020.

[58] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," 2019, *arXiv: 1912.09893*.

[59] J. D. Gardiner, A. J. Laub, J. J. Amato, and C. B. Moler, "Solution of the sylvester matrix equation AXB T + CXD T= E," *ACM Trans. Math. Softw.*, vol. 18, no. 2, pp. 223–231, 1992.

[60] P. Benner and J. Saak, "Numerical solution of large and sparse continuous time algebraic matrix riccati and lyapunov equations: A state of the art survey," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 32–52, 2013.

[61] R. H. Bartels and G. W. Stewart, "Solution of the matrix equation AX + XB= C [F4]," *Commun. ACM*, vol. 15, no. 9, pp. 820–826, 1972.

[62] T. Penzl, "A cyclic low-rank smith method for large sparse lyapunov equations," *SIAM J. Sci. Comput.*, vol. 21, no. 4, pp. 1401–1418, 1999.

[63] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 5812–5823.

[64] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learni.*, 2020, pp. 1597–1607.

[65] M. Caron et al., "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9650–9660.

[66] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

[67] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv: 1807.03748*.

[68] H. J. Van Waarde, M. K. Camlibel, and H. L. Trentelman, "A distance-based approach to strong target control of dynamical networks," *IEEE Trans. Autom. Control*, vol. 62, no. 12, pp. 6266–6277, Dec. 2017.

[69] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7134–7143.

[70] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Control centrality and hierarchical structure in complex networks," *Plos One*, vol. 7, 2012, Art. no. e44459.

[71] B. Wang, L. Gao, and Y. Gao, "Control range: A controllability-based index for node significance in directed networks," *J. Statist. Mechanics: Theory Experiment*, vol. 2012, no. 04, 2012, Art. no. P04011.

[72] C. Commault and J.-M. Dion, "Input addition and leader selection for the controllability of graph-based systems," *Automatica*, vol. 49, no. 11, pp. 3322–3328, 2013.

Anwar Said received the master's degree in computer science from Quaid-I-Azam University, Islamabad, Pakistan, in 2016, and the PhD degree in computer science from Information Technology University, Lahore, in 2021. He is currently a research scientist with Institute for Software Integrated Systems, Vanderbilt University, TN, USA. Prior to this role, he was a post-doctoral research scholar with Vanderbilt University. His research interests include primarily intersects the fields of data science and machine learning, with a focus on graph representation learning, social network analysis, and graph theory with applications in Anhedonia detection, drug discovery and development, and social networks and education. He actively participates in top conferences and journals in these domains, contributing through both publications and serving as a PC member and a reviewer.

Obaid Ullah Ahmad received the BS degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 2019, and the MS degree in computer science from Information Technology University, Lahore, Pakistan, in 2021. He is currently working towards the PhD degree in electrical engineering with the University of Texas at Dallas, Richardson, TX, USA. He is currently a research assistant with the University of Texas at Dallas' Control, Intelligence, Resilience in Networks and Systems Lab, Richardson, TX, USA. His current research interests include control-based approaches, for graph machine learning, network optimization, and multi-agent robot systems.

Waseem Abbas (Member, IEEE) received the MSc and PhD degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, in 2010 and 2013, respectively. He is an assistant professor with the System Engineering Department, University of Texas at Dallas, TX, USA. Previously, he was a research assistant professor with the Vanderbilt University, Nashville, TN, USA. He was a Fulbright scholar from 2009 till 2013. His research interests include the areas of control of networked systems, resilience and robustness in networks, distributed optimization, graph machine learning, and graph-theoretic methods in complex networks.

Mudassir Shabbir received the PhD degree from the Division of Computer Science, Rutgers University, NJ, USA, in 2014. He is an associate professor with the Department of Computer Science, Information Technology University, Lahore, Pakistan and a research assistant professor with Vanderbilt University, Nashville TN. Previously, he has worked at the Lahore University of Management Sciences, Pakistan, Los Alamos National Labs, NM, Bloomberg L.P. New York, NY and with Rutgers University. He was a Rutgers Honors fellow from 2011 to 2012. His research interests include algorithmic and discrete geometry and has developed new methods for the characterization and computation of succinct representations of large data sets with applications in non-parametric statistical analysis. He also works in Graph Machine Learning and Resilient Network Systems.

Xenofon Koutsoukos received the PhD degree in electrical engineering from the University of Notre Dame, in 2000. He is a professor and the chair with the Department of Computer Science and a senior research scientist with the Institute for Software Integrated Systems (ISIS), Vanderbilt University, Nashville, TN, USA. He was a member of research staff with the Xerox Palo Alto Research Center (PARC) (2000–2002). His research work is in the area of cyber-physical systems with emphasis on learning-enabled systems, formal methods, distributed algorithms, security and resilience, diagnosis and fault tolerance, and adaptive resource management. He has published more than 300 journal and conference papers and he is a co-inventor of four US patents. He was a recipient of the NSF Career Award in 2004, the Excellence in Teaching Award in 2009 from the Vanderbilt University School of Engineering, and the 2011 NASA Aeronautics Research Mission Directorate (ARMD) Associate Administrator (AA) Award in Technology and Innovation. He was named a Fellow of the IEEE for his contributions to the design of resilient cyber-physical systems.